Alarms Reset Considerations

Robert Goodwin
Thu, Oct 1, 2009

Linac alarm scan

The Linac control system has provided alarm scanning support of all channels at 15 Hz ever since the first distributed front end architecture was installed in 1982. A design requirement was to be able to inhibit beam on the next pulse if any of a selected set of channels was found to be out of tolerance. To that end, a beam inhibit bit is included in the alarm flags word of each channel. This note describes some considerations that need to be understood by Linac users, especially including the action of an alarms reset, normally initiated by an Acnet "Big Clear" message. But first, examine a recently recognized problem and suggested solutions:

Problem

If Linac beam is currently inhibited at a time when an alarms reset is performed, there is a danger that a channel whose beam inhibit flag is set will cause a delay in reasserting the beam inhibit control line until its tries_now count runs out. This has been observed recently to allow a beam pulse to be accelerated even when a channel is out-of-tolerance. We need to fix this.

Solution 1

One solution is a change in the logic so that during the alarms reset, a channel that is in the Bad state will have its tries_now counter set to 1, even if the value of tries_needed for that channel is > 1. This would eliminate the above delay in reasserting the beam inhibit control line. One wrinkle for this solution is that, for the case of a channel for which the first alarm scan after an alarms reset is Good, this tries_now count remains at 1, until the first cycle on which a Bad condition is found. This serves to eliminate the filtering action for that channel the first time a Bad condition is found following an alarms reset. As for why one would expect this to occur, one can easily imagine such a case due to the hysteresis logic associated with analog alarm scanning. While a channel is in the Bad state, its reading must be found to be within half a tolerance of the nominal value before it can be accepted as Good.

Solution 2

Another solution is to use a new counter that counts down the cycles following an alarms reset, assuming that the beam is currently inhibited at that time. During each alarm scan, a count of channels having the beam inhibit flag set that are found to be Bad is accumulated. After the alarm scan, this count is checked, and if it is nonzero, the beam inhibit control line is asserted; otherwise, it is lifted. If we also check this new counter that counts down to zero for each cycle following an alarms reset, and if that counter is nonzero, use it as an additional determinant for inhibiting beam. This would then delay the time when the beam inhibit can be lifted following an alarms reset. If we set this new counter to 16, which is the maximum tries_needed supported, then this delay is only about 1 second, meaning that an alarms reset action that hopes to remove a beam inhibit may take a second before it is actually removed. This allows enough time for the confirmation of Bad conditions implied by the tries_needed values.

Caveat

There are two implications of using a tries_needed count > 1 for a given channel. First, it naturally causes a delay in emitting a message about a Bad condition. But more importantly, for a channel that has the beam inhibit flag set, it delays asserting beam inhibit. If a channel is so important as to inhibit Linac beam when in a Bad state, is it advisable to specify tries_needed > 1?

One motivation for setting tries_needed > 1 might be to avoid alarm messages for a channel that occasionally has a Bad state, say, because its reading has some noise. If the reading is occasionally Bad, but the hardware is actually ok, then using tries_needed = 2 may be valid. But might one

rather want to improve the reliability of its reading? After all, how can a 15 Hz Linac control system work well if its readings cannot be believed for every 15 Hz cycle? In the early days, the alarm system served to help us become aware of unreliable readings.

Consider another case. The low energy tank quad power supplies are set to inhibit beam, and they have tries_needed values = 2. If one of these pulsed supplies occasionally misfires, then one will not get a message about it nor have Linac beam inhibited because of it. But that does not mean that all is well. Every time it misfires, beam was accelerated through the Linac with a Bad power supply reading. More significantly, this occurs without any notification by the alarm system that things are amiss. If one used tries_needed = 1, then one would see such occurrences, and perhaps be able to focus on what hardware needs to be looked at for the next shutdown. The price would be that beam would be inhibited on the cycle following a misfired power supply. But this may be a price worth paying.

Alarm scan at 15 Hz

Almost all Linac data is updated in the local data pool at 15 Hz, refreshed with new readings each cycle starting about 1 ms following the Linac beam time. But for a few signals, the hardware may be refreshed less often than 15 Hz, including data obtained from a PLC at a more leisurely rate, such as 2 Hz or 4 Hz. For such cases, the Linac alarm scan is still done at 15 Hz, based upon the latest readings residing in the data pool. If one needs to use tries_needed > 1 for such slower devices, one may want to take this slower refresh rate into account.

Conclusion

The complete Linac alarm scan operates on every Linac 15 Hz cycle and can be expected to help avoid unnecessary loss of beam. (With current PowerPC-based front ends, this alarm scan takes less than 1 ms.) Although it cannot prevent a beam pulse from being accelerated under bad conditions, it can prevent it from being accelerated on the following cycles, as long as any channel remains in the Bad state. But this statement of immediate reaction to inhibit beam pulses cannot be taken literally for a channel that specifies tries_needed > 1. For this reason, careful determination of such cases should be made, with full understanding of the implications stated here.

Details

The maximum number of tries_needed that can be specified is 16. This value resides in the alarms flag word, occupying the low 4 bits. For tries_needed = 1, this nibble is `0x0`, for tries_needed = 2, it is `0x1`, *etc*. For the maximum tries_needed = 16, the nibble is `0xF`. The corresponding counter is also a 4-bit field, which occupies the upper 4 bits of the trip count word, with the lower 12 bits holding the trip count itself, a count of all good-bad or bad-good transitions. Because of this, the number of trips is half this count value, which limits it to 2047.

During the alarm scan, if the tries_needed counter is to be incremented, the above nibble is actually decremented, and if doing so causes it to go negative, it has reached the full count.

The alarm scan not only scans for analog channels, but it also scans binary bits. The latter can output Classic protocol alarm messages, but these are not passed to Acnet's `AEOLUS` alarm handler. To support digital alarms for Acnet, one must use the "combined binary status word" scheme, in which readings can be built comprising bits of status sampled from here and there in the digital data pool. These readings are placed in the analog data pool, with a special alarms flag bit denoting that alarm checking will use digital pattern matching logic, not numerical comparison.